FORTH — 79

A PUBLICATION OF THE FORTH STANDARDS TEAM

October 1980

Produced by: FORTH Standards Team

Distributed by: FORTH Interest Group

FORTH INTEREST GROUP ——— P.O. Box 1105 ——— San Carlos, CA 94070

FORTH-79 STANDARD

A PUBLICATION OF THE FORTH STANDARDS TEAM

CONTENTS

		Page
0.	FOREWORD	1
1.	PURPOSE	1
2.	SCOPE	1
3.	ORGANIZATION	1
4.	DEFINITION OF TERMS	2
5.	REFERENCES	9
6.	REQUIREMENTS	10
7.	COMPLIANCE AND LABELING	11
8.	USE	12
9.	GLOSSARY NOTATION	13
10.	REQUIRED WORD SET	15
11.	EXTENSION WORD SETS	32
	11.1 Double Number Word Set	32
	11.2 Assembler Word Set	34
12.	EXPERIMENTAL PROPOSALS	34

REFERENCE WORD SET

FORTH-79 HANDY REFERENCE CARD

FORTH-79 STANDARD

A PUBLICATION OF THE FORTH STANDARDS TEAM

0. FOREWORD

The computer language FORTH was created by Mr. Charles Moore, as an extensible, multi-level environment containing elements of an operating system, a machine monitor, and facilities for program development and testing.

This Standard is a direct descendant of FORTH-77, a work of the FORTH Users Group (Europe). The constituency of the Standards Team has steadily broadened, to include users of an increasing variety of host computers.

1. PURPOSE

The purpose of this FORTH Standard is to allow transportability of standard FORTH programs in source form among standard FORTH systems. A standard program shall execute equivalently on all standard FORTH systems.

2. SCOPE

This standard shall apply to any Standard FORTH program executing on any Standard FORTH system, provided sufficient computer resources (memory, mass storage) are available.

3. ORGANIZATION

This standard consists of:

1) General Text
2) Definitions of Terms
3) Required Word Set
4) Extension Word Sets

Word sets may be subdivided for conceptual purposes by function:

Nucleus
Interpreter
Compiler
Devices

Tradeoffs

When conflicting choices must be made, the following order shall guide the Standards Team.

1) Functional correctness
   - known bounds, non-ambiguous.

2) Portability
   - repeatable results when transported among Standard systems.

3) Simplicity.

4) Naming clarity
   - uniformity of expression. Descriptive names are preferred over procedural. (i.e., [COMPILE] rather than 'C, and ALLOT rather than DP+! .)

5) Generality.

6) Execution speed.

7) Memory compactness.

8) Compilation speed.

9) Historical continuity.

10) Pronounceability.

11) Teachability.

4. DEFINITIONS OF TERMS

These definitions, when in lower case, are terms used within this Standard. They present terms as specifically used within FORTH.

address, byte

An unsigned number that locates an 8-bit byte in a standard FORTH address space over {0..65,535}. It may be a native machine address or a representation on a virtual machine, locating the 'addr-th' byte within the virtual byte address space. Address arithmetic is modulo 65,536 without overflow.

address, compilation

The numerical value equivalent to a FORTH word definition, which is compiled for that definition. The address interpreter uses this value to locate the machine code corresponding to each definition. (May also be called the code field address.)

address, native machine

The natural address representation of the host computer.

address, parameter field

The address of the first byte of memory associated with a word definition for the storage of compilation addresses (in a colon-definition), numeric data and text characters.

arithmetic

All integer arithmetic is performed with signed 16 or 32 bit two's complement results, unless noted.

block

The unit of data from mass storage, referenced by block number. A block must contain 1024 bytes regardless of the minimum data unit read/written from mass storage. The translation from block number to device and physical record is a function of the implementation.

block buffer

A memory area where a mass storage block is maintained.

byte

An assembly of 8 bits. In reference to memory, it is the storage capacity for 8 bits.

cell

A 16-bit memory location. The n-th cell contains the 2n-th and (2n+1)-th byte of the FORTH address space. The byte order is presently unspecified.

character

A 7-bit number which represents a terminal character. The ASCII character set is considered standard. When contained in a larger field, the higher order bits are zero.

compilation

The action of accepting text words from the input stream and placing corresponding compilation addresses in a new dictionary entry.

defining word

A word that, when executed, creates a new dictionary entry. The new word name is taken from the input stream. If the input stream is exhausted before the new name is available, an error condition exists. Common

defining words are:

: CONSTANT CREATE

definition

See 'word definition'.

dictionary

A structure of word definitions in a computer memory. In systems with a text interpreter, the dictionary entries are organized in vocabularies to enable location by name. The dictionary is extensible, growing toward high memory.

equivalent execution

For the execution of a standard program, a set of non-time dependent inputs will produce the same non-time dependent outputs on any FORTH Standard System with sufficient resources to execute the program. Only standard source code will be transportable.

error condition

An exceptional condition which requires action by the system other than the expected function. Actions may be:

1. ignore, and continue
2. display a message
3. execute a particular word
4. interpret a block
5. return control to the text interpreter

A Standard System shall be provided with a tabulation of the action taken for all specified error conditions.
General error conditions:

1. input stream exhausted before a required <name>.
2. empty stack and full stack for the text interpreter.
3. an unknown word, not a valid number for the text interpreter.
4. compilation of incorrectly nested conditionals.
5. interpretation of words restricted to compilation.

6. FORGETing within the system to a point that removes a word required for correct execution.

7. insufficient space remaining in the dictionary.

false

A zero number represents the false condition flag.

flag

A number that may have two logical states, zero and non-zero. These are named 'true' = non-zero, and 'false' = zero. Standard word definitions leave 1 for true, 0 for false.

glossary

A set of word definitions given in a natural language describing the corresponding computer execution action.

immediate word

A word defined to automatically execute when encountered during compilation, which handles exception cases to the usual compilation. See IF LITERAL ." etc.

input stream

A sequence of characters available to the system, for processing by the text interpreter. The input stream conventionally may be taken from a terminal (via the terminal input buffer) and mass storage (via a block buffer). >IN and BLK specify the input stream. Words using or altering >IN and BLK are responsible for maintaining and restoring control of the input stream.

interpreter, address

The (set of) word definitions which interprets (sequences of) FORTH compilation addresses by executing the word definition specified for each one.

interpreter, text

The (set of) word definitions that repeatedly accepts a word name from the input stream, locates the corresponding dictionary entry, and starts the address interpreter to execute it. Text in the input stream interpreted as a number leaves the corresponding value on the data stack. When in the compile mode, the addresses of FORTH words are compiled into the dictionary for later interpretation by the address interpreter. In this case, numbers are compiled, to be placed on the data stack when later interpreted. Numbers shall be accepted unsigned or negatively signed, according to BASE.

load

The acceptance of text from a mass storage device and execution of the dictionary definition of the words encountered. This is the general method for compilation of new definitions into the dictionary.

mass storage

Data is read from mass storage in the form of 1024 byte blocks. This data is held in block buffers. When indicated as UPDATEd (modified) data will be ultimately written to mass storage.

number

When values exist within a larger field, the high order bits are zero. When stored in memory the byte order of a number is unspecified.

type	range	minimum field
bit	0..1	1
character	0..127	7
byte	0..255	8
number	-32,768..32,767	16
positive number	0..32,767	16
unsigned number	0..65,535	16
double number	-2,147,483,648.. 2,147,483,647	32
positive double number	0..2,147,483,647	32
unsigned double number	0..4,294,967,295	32

When represented on the stack, the higher 16-bits (with sign) of a double number are most accessible. When in memory the higher 16-bits are at the lower address. Storage extends over four bytes toward high memory. The byte order within each 16-bit field is unspecified.

output, pictured

The use of numeric output primitives, which convert numerical values into text strings. The operators are used in a sequence which resembles a symbolic 'picture' of the desired text format. Conversion proceeds from low digit to high, from high memory to low.

program

A complete specification of execution to achieve a specific function (application task) expressed in FORTH source code form.

return

The means of terminating text from the input stream. (Conventionally a null (ASCII 0) indicates end of text in the input stream. This character is left by the 'return' key actuation of the operator's terminal, as an absolute stopper to text interpretation.)

screen

Textual data arranged for editing. By convention, a screen consists of 16 lines (numbered 0 thru 15) of 64 characters each. Screens usually contain program source text, but may be used to view mass storage data. The first byte of a screen occupies the first byte of a mass storage block, which is the beginning point for text interpretation during a load.

source definition

Text consisting of word names suitable for execution by the text interpreter. Such text is usually arranged in screens and maintained on a mass storage device.

stack, data

A last in, first out list consisting of 16-bit binary values. This stack is primarily used to hold intermediate values during execution of word definitions. Stack values may represent numbers, characters, addresses, boolean values, etc.

When the name 'stack' is used, it implies the data stack.

stack, return

A last in, first out list which contains the machine addresses of word definitions whose execution has not been completed by the address interpreter. As a word definition passes control to another definition, the return point is placed on the return stack.

The return stack may cautiously be used for other values, such as loop control parameters, and for pointers for interpretation of text.

string

A sequence of 8-bit bytes containing ASCII characters, located in memory by an initial byte address and byte count.

transportability

This term indicates that equivalent execution results when a program is executed on other than the system on which it was created. See 'equivalent execution'.

true

A non-zero value represents the true condition flag. Any non-zero value will be accepted by a standard word as 'true'; all standard words return one when leaving a 'true' flag.

user area

An area in memory which contains the storage for user variables.

variables, user

So that the words of the FORTH vocabulary may be re-entrant (to different users), a copy of each system variable is maintained in the user area.

vocabulary

An ordered list of word definitions. Vocabulary lists are an advantage in reducing dictionary search time and in separating different word definitions that may carry the same name.

word

A sequence of characters terminated by at least one blank (or 'return'). Words are usually obtained via the input stream, from a terminal or mass storage device.

word definition

A named FORTH execution procedure compiled into the dictionary. Its execution may be defined in terms of machine code, as a sequence of compilation addresses or other compiled words. If named, it may be located by specifying this name and the vocabulary in which it is located.

word name

The name of a word definition. Standard names must be distinguished by their length and first thirty-one characters, and may not contain an ASCII null, blank, or 'return'.

word set

A group of FORTH word definitions listed by common characteristics. The standard word sets consist of:

Required Word Set
- Nucleus Words
- Interpreter Words
- Compiler Words
- Device Words

Extension Word Sets
- 32-bit Word Set
- Assembler Word Set

Included as reference material only:
- Reference Word Set

word set, compiler

Words which add new procedures to the dictionary or aid compilation by adding compilation addresses or data structures to the dictionary.

word set, devices

Words which allow access to mass storage and computer peripheral devices.

word set, interpreter

Words which support interpretation of text input from a terminal or mass storage by execution of corresponding dictionary entries, vocabularies, and terminal output.

word set, nucleus

The FORTH words generally defined in machine code that create the stacks and fundamental stack operators (virtual FORTH machine).

word set, reference

This set of words is provided as a reference document only, as a set of formerly standardized words and candidate words for standardization.

word set, required

The minimum words needed to compile and execute all Standard Programs.

word, standard

A named FORTH procedure definition, formally reviewed and accepted by the Standards Team. A serial number identifier {100..999} indicates a Standard Word. A functional alteration of a Standard Word will require assignment of a new serial number identifier.

The serial number identifier has no required use, other than to correlate the definition name with its unique Standard definition.

5. REFERENCES

The following documents are considered to be a portion of this Standard:

American Standard Code for Information Interchange,
American National Standards Institute, X3.4-1968

Webster's Collegiate Dictionary shall be used to resolve conflicts in spelling and English word usage.

The following documents are noted as pertinent to the FORTH-79 Standard, but are not part of this Standard.

FORTH-77, FORTH Users Group, FST-780314

FORTH-78, FORTH International Standards Team

6. REQUIREMENTS

6.1 Documentation Requirements

Each Standard System and Standard Program shall be accompanied by a statement of the minimum (byte) requirements for:

1. System dictionary space
2. Application dictionary space
3. Data stack
4. Return stack
5. Mass storage contiguous block quantity required
6. An operator's terminal.

Each Standard System shall be provided with a statement of the system action upon each of the error conditions as identified in this Standard.

6.2 Testing Requirements

The following host computer configuration is specified as a minimum environment for testing against this Standard. Applications may require different capacities.

1. 2000 bytes of memory for application dictionary
2. Data stack of 64 bytes
3. Return stack of 48 bytes
4. Mass storage capacity of 32 blocks, numbered 0 through 31
5. One ASCII input/output device acting as an operator's terminal.

7. COMPLIANCE AND LABELING

The FORTH Standards Team hereby specifies the requirements for labeling of systems and applications so that the conditions for program portability may be established.

A system may use the specified labeling if it complies with the terms of this Standard, and meets the particular Word Set definitions.

A Standard Program (application) may use the specified labeling if it utilizes the specified standard system according to this Standard, and executes equivalently on any such system.

FORTH Standard

A system may be labeled 'FORTH-79 Standard' if it includes all of the Required Word Set in either source or object form, and complies with the text of this Standard. After executing "79-STANDARD" the dictionary must contain all of the Required Word Set in the vocabulary FORTH, as specified in this Standard.

Standard Sub-set

A system may be labeled 'FORTH-79 Standard Sub-set' if it includes a portion of the Required Word Set, and complies with the remaining text of this standard. However, no Required Word may be present with a non-standard definition.

Standard with Extensions

A system may be labeled 'FORTH-79 Standard with <name> Standard Extension(s)' if it comprises a FORTH-79 Standard System and one or more Standard Extension Word Set(s). The designation would be in the form:

'FORTH-79 Standard with Double-Number Standard Extensions'

8. USE

A FORTH Standard program may reference only the definitions of the Required Word Set, and definitions which are subsequently defined in terms of these words. Furthermore, a FORTH Standard program must use the standard words as required by any conventions of this Standard. Equivalent execution must result from Standard programs.

The FORTH system may share the dictionary space with the user's application, and the native addressing protocol of the host computer is beyond the scope of this Standard.

Therefore, in a Standard program, the user may only operate on data which was stored by the application. No exceptions!

A Standard Program may address:

1. parameter fields of variables, constants and DOES> words. A DOES> word's parameter field may only be addressed with respect to the address left by DOES> , itself.

2. dictionary space ALLOTed.

3. data in mass storage block buffers. (Note restriction in BLOCK on latest buffer addressing.)

4. the user area and PAD.

A Standard Program may NOT address:

1. directly into the data or return stacks.

2. into a definition's name field, link field, or code field.

3. into a definition's parameter field if not stored by the application.

Further usage requirements are expected to be added for transporting programs between standard systems.

FORTH Standard definitions have a serial number assigned, in the range 100 thru 999. Neither a Standard System nor Standard Program may redefine these word names, within the FORTH vocabulary.

9. GLOSSARY NOTATION

Order

The Glossary definitions are listed in ASCII alphabetical order.

Stack Notation

The first line of each entry describes the execution of the definition:

stack parameters before execution
--- showing point of execution
stack parameters after execution

i.e., before --- after

In this notation, the top of the stack is to the right. Words may also be shown in context, when appropriate.

Attributes

Capitalized symbols indicate attributes of the defined words:

C The word may only be used within a colon-definition.

I Indicates that the word is IMMEDIATE and will execute during compilation, unless special action is taken.

U A user variable.

Capitalization

Word names as used within the dictionary are conventionally written in upper case characters. Within this Standard lower case will be used when reference is made to the run-time machine code, not directly accessible, i.e., VARIABLE is the user word to create a variable. Each use of that variable makes use of a code sequence 'variable' which executes the function of the particular variable.

Pronunciation

The natural language pronunciation of FORTH names is given in double quotes (").

Stack Parameters

Unless otherwise stated, all references to numbers apply to 16-bit signed integers.

The implied range of values is shown as {from..to}. The content of an address is shown by double curly brackets, particularly for the contents of variables. i.e., BASE {{2..70}}

addr {0..65,535}

A value representing the address of a byte, within the FORTH standard memory space. This addressed byte may represent the first byte of a larger data field in memory.

byte {0..255}

A value representing an 8 bit byte. When in a larger field, the higher bits are zero.

char {0..127}

A value representing a 7 bit ASCII character code. When in a larger field, the higher bits are zero.

d {-2,147,483,648..2,147,483,647}

32 bit signed 'double' number. The most significant 16-bits, with sign, is most accessible on the stack.

flag

A numerical value with two logical states; 0= false, non-zero = true.

n {-32,768..32,767}

16 bit signed integer number.

Any other symbol refers to an arbitrary signed 16-bit integer in the range {-32,768..32,767}, unless otherwise noted.

Input Text

<name>

An arbitrary FORTH word accepted from the input stream. This notation refers to text from the input stream, not to values on the data stack. If the input stream is exhausted before encountering <name>, an error condition exists.

10. REQUIRED WORD SET

The words of the Required Word Set are grouped to show like characteristics. No implementation requirements should be inferred from this grouping.

Nucleus Words

```
!   *   */   */MOD   +   +!   +loop   -   /
/MOD   0<   0=   0>   1+   1-   2+   2-   <
=   >   >R   ?DUP   @   ABS   AND   begin   C!
C@   colon   CMOVE   constant   create   D+
D<   DEPTH   DNEGATE   do   does>
DROP   DUP   else   EXECUTE   EXIT   FILL   I
if   J   LEAVE   literal   loop   MAX   MIN
MOD   MOVE   NEGATE   NOT   OR   OVER   PICK
R>   R@   repeat   ROLL   ROT   semicolon
SWAP   then   U*   U/   U<   until   variable
while   XOR
```

(note that the lower case entries refer to just the run-time code corresponding to a compiling word.)

Interpreter Words

```
#   #>   #S   '   (   -TRAILING   .
79-STANDARD   <#   >IN   ?   ABORT   BASE   BLK
CONTEXT   CONVERT   COUNT   CR   CURRENT
DECIMAL   EMIT   EXPECT   FIND   FORTH   HERE
HOLD   KEY   PAD   QUERY   QUIT   SIGN   SPACE
SPACES   TYPE   U.   WORD
```

Compiler Words

```
+LOOP   ,   ."   :   ;   ALLOT   BEGIN
COMPILE   CONSTANT   CREATE   DEFINITIONS   DO
DOES>   ELSE   FORGET   IF   IMMEDIATE
LITERAL   LOOP   REPEAT   STATE   THEN   UNTIL
VARIABLE   VOCABULARY   WHILE   [   [COMPILE]   ]
```

Device Words

```
BLOCK   BUFFER   EMPTY-BUFFERS   LIST
LOAD   SAVE-BUFFERS   SCR   UPDATE
```

! n addr —— 112

Store n at address. "store"

ud1 —— ud2 158

Generate from an unsigned double-number d1, the next ASCII character which is placed in an output string. Result d2 is the quotient after division by BASE and is maintained for further processing. Used between <# and #>. "sharp"

#> d —— addr n 190

End pictured numeric output conversion. Drop d, leaving the text address, and character count, suitable for TYPE. "sharp-greater"

#S ud —— 0 0 209

Convert all digits of an unsigned 32-bit number ud, adding each to the pictured numeric output text, until remainder is zero. A single zero is added to the output string if the number was initially zero. Use only between <# and #>. "sharp-s"

' —— addr I,171

Used in the form:

' <name>

If executing, leave the parameter field address of the next word accepted from the input stream. If compiling, compile this address as a literal; later execution will place this value on the stack. An error condition exists if not found after a search of the CONTEXT and FORTH vocabularies. Within a colon-definition ' <name> is identical to [' <name>] LITERAL. "tick"

(I,122

Used in the form:

(ccc)

Accept and ignore comment characters from the input stream, until the next right parenthesis. As a word, the left parenthesis must be followed by one blank. It may be freely used while executing or compiling. An error condition exists if the input stream is exhausted before the right parenthesis. "paren" The right parenthesis is pronounced "close-paren"

* n1 n2 —— n3 138

Leave the arithmetic product of n1 times n2. "times"

* n1 n2 -- n3 138

Leave the arithmetic product of n1 times n2. "times"

*/ n1 n2 n3 -- n4 220

Multiply n1 by n2 , divide the result by n3 and leave the quotient n4. n4 is rounded toward zero. The product of n1 times n2 is maintained as an intermediate 32-bit value for greater precision than the otherwise equivalent sequence: n1 n2 * n3 / "times-divide"

*/MOD n1 n2 n3 -- n4 n5 192

Multiply n1 by n2, divide the result by n3 and leave the remainder n4 and quotient n5. A 32-bit intermediate product is used as for */. The remainder has the same sign as n1. "times-divide-mod"

+ n1 n2 -- n3 121

Leave the arithmetic sum of n1 plus n2. "plus"

+! n addr -- 157

Add n to the 16-bit value at the address, by the convention given for +. "plus-store"

+LOOP n -- I,C,141

Add the signed increment n to the loop index using the convention for +, and compare the total to the limit. Return execution to the corresponding DO until the new index is equal to or greater than the limit (n>0), or until the new index is less than the limit (n<0). Upon the exiting from the loop, discard the loop control parameters, continuing execution ahead. Index and limit are signed integers in the range {-32,768..32,767}. "plus-loop"

(Comment: It is a historical precedent that the limit for n<0 is irregular. Further consideration of the characteristic is likely.)

, n -- 143

Allot two bytes in the dictionary, storing n there. "comma"

- n1 n2 -- n3 134

Subtract n2 from n1 and leave the difference n3. "minus"

-TRAILING addr n1 -- addr n2 148

Adjust the character count n1 of a text string beginning at addr to exclude trailing blanks, i.e., the characters at addr+n2 to addr+n1-1 are blanks. An error condition exists if n1 is negative. "dash-trailing"

. n — 193

Display n converted according to BASE in a free-field format with one trailing blank. Display only a negative sign. "dot"

." I,133

Interpreted or used in a colon-definition in the form:

." cccc"

Accept the following text from the input stream, terminated by " (double-quote). If executing, transmit this text to the selected output device. If compiling, compile so that later execution will transmit the text to the selected output device. At least 127 characters are allowed in the text. If the input stream is exhausted before the terminating double-quote, an error condition exists. "dot-quote"

/ n1 n2 — n3 178

Divide n1 by n2 and leave the quotient n3. n3 is rounded toward zero. "divide"

/MOD n1 n2 — n3 n4 198

Divide n1 by n2 and leave the remainder n3 and quotient n4. n3 has the same sign as n1. "divide-mod"

0< n — flag 144

True if n is less than zero (negative). "zero-less"

0= n — flag 180

True if n is zero. "zero-equals"

0> n — flag 118

True if n is greater than zero. "zero-greater"

1+ n — n+1 107

Increment n by one, according to the operation for +. "one-plus"

1- n — n-1 105

Decrement n by one, according to the operation -. "one-minus"

2+ n — n+2 135

Increment n by two, according to the operation for +. "two-plus"

2- • n —— n-2 129

Decrement n by two, according to the operation for -. "two-minus"

79-STANDARD 119

Execute assuring that a FORTH-79 Standard system is available, otherwise an error condition exists.

: 116

A defining word used in the form:

: <name> . . . ;

Select the CONTEXT vocabulary to be identical to CURRENT. Create a dictionary entry for <name> in CURRENT, and set compile mode. Words thus defined are called 'colon-definitions'. The compilation addresses of subsequent words from the input stream which are not immediate words are stored into the dictionary to be executed when <name> is later executed. IMMEDIATE words are executed as encountered.

If a word is not found after a search of the CONTEXT and FORTH vocabularies, conversion and compilation of a literal number is attempted, with regard to the current BASE; that failing, an error condition exists. "colon"

; I,C,196

Terminate a colon-definition and stop compilation. If compiling from mass storage and the input stream is exhausted before encountering ; an error condition exists. "semi-colon"

< n1 n2 —— flag 139

True if n1 is less than n2.

-32768 32767 < must return true.
-32768 0 must be distinguished. "less-than"

<# 169

Initialize pictured numeric output. The words:

<# # #S HOLD SIGN #>

can be used to specify the conversion of a double-precision number into an ASCII character string stored in right-to-left order. "less-sharp"

= n1 n2 —— flag 173

True if n1 is equal to n2. "equals"

`>` n1 n2 -- flag 102

True if n1 is greater than n2. "greater-than"

`>IN` -- addr U,201

Leave the address of a variable which contains the present character offset within the input stream {{0..1023}} "to-in"

See: WORD (." FIND

`>R` n -- C,200

Transfer n to the return stack. Every >R must be balanced by a R> in the same control structure nesting level of a colon-definition. "to-r"

`?` addr -- 194

Display the number at address, using the format of ".". "question-mark"

`?DUP` n -- n (n) 184

Duplicate n if it is non-zero. "query-dup"

`@` addr -- n 199

Leave on the stack the number contained at addr. "fetch"

`ABORT` 101

Clear the data and return stacks, setting execution mode. Return control to the terminal.

`ABS` n1 -- n2 108

Leave the absolute value of a number. "absolute"

`ALLOT` n -- 154

Add n bytes to the parameter field of the most recently defined word.

`AND` n1 n2 -- n3 183

Leave the bitwise logical 'and' of n1 and n2.

`BASE` -- addr U,115

Leave the address of a variable containing the current input-output numeric conversion base. {{2..70}}

BEGIN I,C,147

Used in a colon-definition in the forms:

```
BEGIN . . . flag UNTIL              or
BEGIN . . . flag WHILE ... REPEAT
```

BEGIN marks the start of a word sequence for repetitive execution. A BEGIN-UNTIL loop will be repeated until flag is true. A BEGIN-WHILE-REPEAT loop will be repeated until flag is false. The words after UNTIL or REPEAT will be executed when either loop is finished. flag is always dropped after being tested.

BLK —— addr U,132

Leave the address of a variable containing the number of the mass storage block being interpreted as the input stream.
If the content is zero, the input stream is taken from the terminal.
"b-l-k" {{unsigned-number}}

BLOCK n —— addr 191

Leave the address of the first byte in block n. If the block is not already in memory, it is transferred from mass storage into whichever memory buffer has been least recently accessed. If the block occupying that buffer has been UPDATEd (i.e. modified), it is rewritten onto mass storage before block n is read into the buffer. n is an unsigned number. If correct mass storage read or write it not possible, an error condition exists. Only data within the latest block referenced by BLOCK is valid by byte address, due to sharing of the block buffers.

BUFFER n —— addr 130

Obtain the next block buffer, assigning it to block n. The block is not read from mass storage. If the previous contents of the buffer has been marked as UPDATEd, it is written to mass storage. If correct writing to mass storage is not possible, an error condition exists. The address left is the first byte within the buffer for data storage. n is an unsigned number.

C! n addr —— 219

Store the least significant 8-bits of n at addr. "c-store"

C@ addr —— byte 156

Leave on the stack the contents of the byte at addr (with higher bits zero, in a 16-bit field). "c-fetch"

CMOVE addr1 addr2 n —— 153

Move n bytes beginning at address addr1 to addr2. The contents of addr1 is moved first proceeding toward high memory. If n is zero or negative nothing is moved. "c-move"

COMPILE C,146

When a word containing COMPILE executes, the 16-bit value following the compilation address of COMPILE is copied (compiled) into the dictionary. i.e., COMPILE DUP will copy the compilation address of DUP.

COMPILE [0 ,] will copy zero.

CONSTANT n —— 185

A defining word used in the form:

n CONSTANT <name>

to create a dictionary entry for <name>, leaving n in its parameter field. When <name> is later executed, n will be left on the stack.

CONTEXT —— addr U,151

Leave the address of a variable specifying the vocabulary in which dictionary searches are to made, during interpretation of the input stream.

CONVERT d1 addr1 —— d2 addr2 195

Convert to the equivalent stack number the text beginning at addr1+1 with regard to BASE. The new value is accumulated into double number d1, being left as d2. addr2 is the address of the first non-convertible character.

COUNT addr —— addr+1 n 159

Leave the address addr+1 and the character count of text beginning at addr. The first byte at addr must contain the character count n. Range of n is {0..255}.

CR 160

Cause a carriage-return and line-feed to occur at the current output device. "c-r"

CREATE 239

A defining word used in the form:

CREATE <name>

to create a dictionary entry for <name>, without allocating any parameter field memory. When <name> is subsequently executed, the address of the first byte of <name>'s parameter field is left on the stack.

CURRENT —— addr U,137

Leave the address of a variable specifying the vocabulary into which new word definitions are to be entered.

D+ d1 d2 —— d3 241

Leave the arithmetic sum of d1 plus d2. "d-plus"

D< d1 d2 —— flag 244

True if d1 is less than d2. "d-less-than"

DECIMAL 197

Set the input-output numeric conversion base to ten.

DEFINITIONS 155

Set CURRENT to the CONTEXT vocabulary so that subsequent definitions will be created in the vocabulary previously selected as CONTEXT.

DEPTH —— n 238

Leave the number of the quantity of 16-bit values contained in the data stack, before n was added.

DNEGATE d —— -d 245

Leave the two's complement of a double number.

DO n1 n2 —— I,C,142

Use in a colon-definition:

DO . . . LOOP or
DO . . . +LOOP

Begin a loop which will terminate based on control parameters. The loop index begins at n2, and terminates based on the limit n1. At LOOP or +LOOP, the index is modified by a positive or negative value. The range of a DO-LOOP is determined by the terminating word.

DO-LOOP may be nested. Capacity for three levels of nesting is specified as a minimum for standard systems.

DOES> I,C,168

Define the run-time action of a word created by a high-level defining word. Used in the form:

: <name> . . . CREATE . . . DOES> . . . ;

and then <name> <namex>

Marks the termination of the defining part of the defining word <name> and begins the definition of the run time action for words that will later be defined by <name>. On execution of <namex> the sequence of words between DOES> and ; will be executed, with the address of <namex>'s parameter field on the stack. "does"

DROP n — 233

Drop the top number from the stack.

DUP n — n n 205

Leave a copy of the top stack number.

ELSE I,C,167

Used in a colon-definition in the form:

IF . . . ELSE . . . THEN

ELSE executes after the true part following IF. ELSE forces execution to skip till just after THEN. It has no effect on the stack. (See IF)

EMIT char — 207

Transmit character to the current output device.

EMPTY-BUFFERS 145

Mark all block buffers as empty, without necessarily affecting their actual contents. UPDATEd blocks are not written to mass storage.

EXECUTE addr — 163

Execute the dictionary entry whose compilation address is on the stack.

EXIT C,117

When compiled within a colon-definition, terminate execution of that definition, at that point. May not be used within a DO...LOOP.

EXPECT addr n — 189

Transfer characters from the terminal beginning at addr, upward, until a "return" or the count of n has been received. Take no action for n less than or equal to zero. One or two nulls are added at the end of text.

FILL addr n byte — 234

Fill memory beginning at address with a sequence of n copies of byte. If the quantity n is less than or equal to zero, take no action.

FIND — addr 203

Leave the compilation address of the next word name, which is accepted from the input stream. If that word cannot be found in the dictionary after a search of CONTEXT and FORTH leave zero.

FORGET 186

Execute in the form:

FORGET <name>

Delete from the dictionary <name> (which is in the CURRENT vocabulary) and all words added to the dictionary after <name>, regardless of their vocabulary. Failure to find <name> in CURRENT or FORTH is an error condition.

FORTH I,187

The name of the primary vocabulary. Execution makes FORTH the CONTEXT vocabulary.

New definitions become a part of FORTH until a differing CURRENT vocabulary is established.

User vocabularies conclude by 'chaining' to FORTH, so it should be considered that FORTH is 'contained' within each user's vocabulary.

HERE -- addr 188

Return the address of the next available dictionary location.

HOLD char -- 175

Insert char into a pictured numeric output string. May only be used between <# and #>.

I -- n C,136

Copy the loop index onto the data stack. May be only used in the form:

DO . . . I . . . LOOP or DO . . . I . . . +LOOP

IF flag -- I,C,210

Used in a colon-definition in the forms:

flag IF . . . ELSE . . . THEN or
flag IF . . . THEN

If flag is true, the words following IF are executed and the words following ELSE are skipped. The ELSE part is optional.

If flag is false, words between IF and ELSE, or between IF and THEN (when no ELSE is used), are skipped. IF-ELSE-THEN conditionals may be nested.

IMMEDIATE 103

Mark the most recently made dictionary entry as a word which will be executed when encountered during compilation rather than compiled.

J —— n C,225

Return the index of the next outer loop. May be used only within a nested DO-LOOP in the form:

DO . . . DO . . . J . . . LOOP . . . LOOP

KEY —— char 100

Leave the ASCII value of the next available character from the current input device.

LEAVE C,213

Force termination of a DO-LOOP at the next LOOP or +LOOP by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until the loop terminating word is encountered.

LIST n —— 109

List the ASCII symbolic contents of screen n on the current output device, setting SCR to contain n. n is unsigned.

LITERAL n —— I,215

If compiling, then compile the stack value n as a 16-bit literal, which when later executed, will leave n on the stack.

LOAD n —— 202

Begin interpretation of screen n by making it the input stream; preserve the locators of the present input stream (from >IN and BLK). If interpretation is not terminated explicitly it will be terminated when the input stream is exhausted. Control then returns to the input stream containing LOAD, determined by the input stream locators >IN and BLK.

LOOP I,C,124

Increment the DO-LOOP index by one, terminating the loop if the new index is equal to or greater than the limit. The limit and index are signed numbers in the range {-32,768..32,767}.

MAX n1 n2 —— n3 218

Leave the greater of two numbers. "max"

MIN n1 n2 —— n3 127

Leave the lesser of two numbers. "min"

MOD n1 n2 -- n3 104

Divide n1 by n2, leaving the remainder n3, with the same sign as n1. "mod"

MOVE addr1 addr2 n -- 113

Move the specified quantity n of 16-bit memory cells beginning at addr1 into memory at addr2. The contents of addr1 is moved first. If n is negative or zero, nothing is moved.

NEGATE n -- -n 177

Leave the two's complement of a number, i.e., the difference of 0 less n.

NOT flag1 -- flag2 165

Reverse the boolean value of flag1. This is identical to 0=.

OR n1 n2 -- n3 223

Leave the bitwise inclusive-or of two numbers.

OVER n1 n2 -- n1 n2 n1 170

Leave a copy of the second number on the stack.

PAD -- addr 226

The address of a scratch area used to hold character strings for intermediate processing. The minimum capacity of PAD is 64 characters (addr through addr+63).

PICK n1 -- n2 240

Return the contents of the n1-th stack value, not counting n1 itself. An error condition results for n less than one.

2 PICK is equivalent to OVER. {1 .. n}

QUERY 235

Accept input of up to 80 characters (or until a 'return') from the operator's terminal, into the terminal input buffer. WORD may be used to accept text from this buffer as the input stream, by setting >IN and BLK to zero.

QUIT 211

Clear the return stack, setting execution mode, and return control to the terminal. No message is given.

R> —— n C,110

Transfer n from the return stack to the data stack. "r-from"

R@ —— n C,228

Copy the number on the top of the return stack to the data stack. "r-fetch"

REPEAT I,C,120

Used in a colon-definition in the form:

BEGIN ... WHILE ... REPEAT

At run-time, REPEAT returns to just after the corresponding BEGIN.

ROLL n —— 236

Extract the n-th stack value to the top of the stack, not counting n itself, moving the remaining values into the vacated position. An error condition results for n less than one. {1 .. n}

3 ROLL = ROT

1 ROLL = null operation

ROT n1 n2 n3 —— n2 n3 n1 212

Rotate the top three values, bringing the deepest to the top. "rote"

SAVE-BUFFERS 221

Write all blocks to mass-storage that have been flagged as UPDATEd. An error condition results if mass-storage writing is not completed.

SCR —— addr U,217

Leave the address of a variable containing the number of the screen most recently listed. "s-c-r" unsigned-number

SIGN n —— C,140

Insert the ASCII "-" (minus sign) into the pictured numeric output string, if n is negative.

SPACE 232

Transmit an ASCII blank to the current output device.

SPACES n — 231

Transmit n spaces to the current output device. Take no action for n of zero or less.

STATE — addr U,164

Leave the address of the variable containing the compilation state. A non-zero content indicates compilation is occurring, but the value itself may be installation dependent.

SWAP n1 n2 — n2 n1 230

Exchange the top two stack values.

THEN I,C,161

Used in a colon-definition, in the form:

IF . . . ELSE . . . THEN or
IF . . . THEN

THEN is the point where execution resumes after ELSE or IF (when no ELSE is present).

TYPE addr n — 222

Transmit n characters beginning at address to the current output device. No action takes place for n less than or equal to zero.

U* un1 un2 — ud3 242

Perform an unsigned multiplication of un1 by un2, leaving the double number product ud3. All values are unsigned. "u-times"

U. un — 106

Display un converted according to BASE as an unsigned number, in a free-field format, with one trailing blank. "u-dot"

U/MOD udl un2 — un3 un4 243

Perform the unsigned division of double number udl by un2, leaving the remainder un3, and quotient un4. All values are unsigned. "u-divide-mod"

U< un1 un2 — flag 150

Leave the flag representing the magnitude comparison of un1 < un2 where un1 and un2 are treated as 16-bit unsigned integers. "u-less-than"

UNTIL flag —— I,C,237

Within a colon-definition, mark the end of a BEGIN-UNTIL loop, which will terminate based on a flag. If flag is true, the loop is terminated. If flag is false, execution returns to the first word after BEGIN. BEGIN-UNTIL structures may be nested.

UPDATE 229

Mark the most recently referenced block as modified. The block will subsequently be automatically transferred to mass storage should its memory buffer be needed for storage of a different block, or upon execution of SAVE-BUFFERS.

VARIABLE 227

A defining word executed in the form:

VARIABLE <name>

to create a dictionary entry for <name> and allot two bytes for storage in the parameter field. The application must initialize the stored value. When <name> is later executed, it will place the storage address on the stack.

VOCABULARY 208

A defining word executed in the form:

VOCABULARY <name>

to create (in the CURRENT vocabulary) a dictionary entry for <name>, which specifies a new ordered list of word definitions. Subsequent execution of <name> will make it the CONTEXT vocabulary. When <name> becomes the CURRENT vocabulary (see DEFINITIONS), new definitions will be created in that list.

In lieu of any further specification, new vocabularies 'chain' to FORTH. That is, when a dictionary search through a vocabulary is exhausted, FORTH will be searched.

WHILE flag —— I,C,149

Used in a colon-definition in the form:

BEGIN ... flag WHILE ... REPEAT

Select conditional execution based on the flag. On a true flag, continue execution through to REPEAT, which then returns back to just after BEGIN. On a false flag, skip execution to just after REPEAT, exiting the structure.

WORD char —— addr 181

Receive characters from the input stream until the non-zero delimiting character is encountered or the input stream is exhausted, ignoring leading delimiters. The characters are stored as a packed string with the character count in the first character position. The actual delimiter encountered (char or null) is stored at the end of the text but not included in the count. If the input stream was exhausted as WORD is called, then a zero length will result. The address of the beginning of this packed string is left on the stack.

XOR n1 n2 —— n3 174

Leave the bitwise exclusive-or of two numbers. "x-or"

[I,125

End the compilation mode. The text from the input stream is subsequently executed. See] "left-bracket"

[COMPILE] I,C,179

Used in a colon-definition in the form:

[COMPILE] <name>

Force compilation of the following word. This allows compilation of an IMMEDIATE word when it would otherwise be executed. "bracket-compile"

] 126

Set the compilation mode. The text from the input stream is subsequently compiled. See ["right-bracket"

11. EXTENSION WORD SETS

11.1 DOUBLE NUMBER WORD SET

2! d addr ——

Store d in 4 consecutive bytes beginning at addr, as for a double number. "two-store"

2@ addr —— d

Leave on the stack the contents of the four consecutive bytes beginning at addr, as for a double number. "two-fetch"

2CONSTANT d ——

A defining word used in the form:

d 2CONSTANT <name>

to create a dictionary entry for <name>, leaving d in its parameter field. When <name> is later executed, d will be left on the stack. "two-constant"

2DROP d ——

Drop the top double number on the stack. "two-drop"

2DUP d —— d d

Duplicate the top double number on the stack. "two-dup"

2OVER d1 d2 —— d1 d2 d1

Leave a copy of the second double number on the stack. "two-over"

2ROT d1 d2 d3 —— d2 d3 d1

Rotate the third double number to the top of the stack. "two-rote"

2SWAP d1 d2 —— d2 d1

Exchange the top two double numbers on the stack. "two-swap"

2VARIABLE

A defining word used in the form:

2VARIABLE <name>

to create a dictionary entry of <name> and assign 4 bytes for storage in the parameter field. When <name> is later executed, it will leave the address of the first byte of its parameter field on the stack. "two-variable"

D+ d1 d2 — d3 241

Leave the arithmetic sum of d1 and d2. "d-plus"

D- d1 d2 — d3

Subtract d2 from d1 and leave the difference d3. "d-minus"

D. d — 129

Display d converted according to BASE in a free-field format, with one trailing blank. Display the sign only if negative. "d-dot"

D.R d n —

Display d converted according to BASE, right aligned in an n character field. Display the sign only if negative. "d-dot-r"

D0= d — flag

Leave true if d is zero. "d-zero-equals"

D< d1 d2 — flag 244

True if d1 is less than d2. "d-less"

D= d1 d2 — flag

True if d1 equals d2. "d-equal"

DABS d1 — d2

Leave as a positive double number d2, the absolute value of a double number, d1. {0..2,147,483,647} "d-abs"

DMAX d1 d2 — d3

Leave the larger of two double numbers. "d-max"

DMIN d1 d2 — d3

Leave the smaller of two double numbers. "d-min"

DNEGATE d — -d 245

Leave the double number two's complement of a double number, i.e., the difference 0 less d. "d-negate"

DU< ud1 ud2 — flag

True of ud1 is less than ud2. Both numbers are unsigned. "d-u-less"

11.2 ASSEMBLER WORD SET

;CODE C,I,206

Used in the form:

: <name> . . . ;CODE

Stop compilation and terminate a defining word <name>. ASSEMBLER becomes the CONTEXT vocabulary. When <name> is executed in the form:

<name> <namex>

to define the new <namex>, the execution address of <namex> will contain the address of the code sequence following the ;CODE in <name>. Execution of any <namex> will cause this machine code sequence to be executed. "semi-colon-code"

ASSEMBLER I,166

Select assembler as the CONTEXT vocabulary.

CODE 111

A defining word used in the form:

CODE <name> . . . END-CODE

to create a dictionary entry for <name> to be defined by a following sequence of assembly language words. ASSEMBLER becomes the context vocabulary.

END-CODE

Terminate a code definition, resetting the CONTEXT vocabulary to the CURRENT vocabulary. If no errors have occurred, the code definition is made available for use.

12. EXPERIMENTAL PROPOSALS

No Experimental Proposals were submitted for publication.

REFERENCE WORD SET

This word set is furnished as a reference document. It is a set of formerly standardized words and candidate words for standardization.

The Reference Word Set contains both Standard Word Definitions (with Serial number identifiers in the range 100 through 999), and uncontrolled word definitions.

Uncontrolled definitions are included for public reference of words that have present usage and/or are candidates for future standardization.

No restrictions are placed on the definition or usage of uncontrolled words. However, use of these names for procedures differing from the given definitions is discouraged.

!BITS n1 addr n2 ——

Store the value of n1 masked by n2 into the equivalent masked part of the contents of addr, without affecting bits outside the mask. "store-bits"

** n1 n2 —— n3

Leave the value of n1 to the power n2. "power"

+BLOCK n1 —— n2

Leave the sum of n1 plus the number of the block being interpreted. n1 and n2 are unsigned. "plus-block"

-' —— (addr) flag

Used in the form:

-' <name>

Leave the parameter field of <name> beneath zero (false) if the name can be found in the CONTEXT vocabulary; leave only true if not found. "dash-tick".

—> I,131

Continue interpretation on the next sequential block. May be used within a colon-definition that crosses a block boundary. "next-block"

-MATCH addr1 n1 addr2 n2
—— addr3 f

Attempt to find the n2-character string beginning at addr2 somewhere in the n1-character string beginning at addr1. Return the last +1 character address addr3 of the match point and a flag which is zero if a match exists. "dash-match"

-TEXT addr1 n1 addr2 -- n2

Compare two strings over the length n1 beginning at addr1 and addr2. Return zero if the strings are equal. If unequal, return n2, the difference between the last characters compared:

addr1(i) - addr2(i)

"dash-text"

.R n1 n2 --

Print n1 right aligned in a field of n2 characters, according to BASE. If n2 is less than 1, no leading blanks are supplied. "dot-r"

/LOOP n --

A DO-LOOP terminating word. The loop index is incremented by the unsigned magnitude of n. Until the resultant index exceeds the limit, execution returns to just after the corresponding DO: otherwise, the index and limit are discarded. Magnitude logic is used. "up-loop"

1+! addr --

Add one to the 16-bit contents at addr. "one-plus-store"

1-! addr --

Subtract 1 from the 16-bit contents at addr. "one-minus-store"

2* n1 -- n2

Leave 2*(n1). "two-times"

2/ n1 -- n2

Leave (n1)/2. "two-divide"

;: C

Used to specify a new defining word:

```
: <name>  . . .
     ;:   . . .  ;
<name>  <namex>
```

When <name> is executed, it creates an entry for the new word <namex>. Later execution of <namex> will execute the sequence of words between ;: and ; , with the address of the first (if any) parameters associated with <namex> on the stack. "semi-colon-colon"

;S

Stop interpretation of a block. For execution only. "semi-s"

<> n1 n2 -- flag

Leave true if n1 is not equal to n2. "not-equal"

<BUILDS C

Used in conjunction with DOES> in defining words, in the form:

```
: <name> . . . <BUILDS . . .
      DOES> . . . ;
```

and then <name> <namex>

When <name> executes, <BUILDS creates a dictionary entry for the new <namex>. The sequence of words between <BUILDS and DOES> established a parameter field for <namex>. When <namex> is later executed, the sequence of words following DOES> will be executed, with the parameter field address of <namex> on the data stack. "builds"

<CMOVE addr1 addr2 n --

Copy n bytes beginning at addr1 to addr2. The move proceeds within the bytes from high memory toward low memory. "reverse-c-move"

>< n1 -- n2

Swap the high and low bytes within n1. "byte-swap"

>MOVE< addr1 addr2 n --

Move n bytes beginning at addr1 to the memory beginning at addr2. During this move, the order of each byte pair is reversed. "byte-swap-move"

@BITS addr n1 -- n2

Return the 16-bits at addr masked by n1. "fetch-bits"

ABORT" flag -- I,C

Used in a colon-definition in the form:

```
ABORT" stack empty"
```

If the flag is true, print the following text, till ". Then execute ABORT. "abort-quote"

AGAIN I,C,114

Effect an unconditional jump back to the start of a BEGIN-AGAIN loop.

ASCII -- char (executing)
       -- (compiling) I,C

Leave the ASCII character value of the next non-blank character in the input stream. If compiling, compile it as a literal, which will be later left when executed.

ASHIFT n1 n2 -- n3

Shift the value n1 arithemetically n2 bits left if n2 is positive, shifting zeros into the least-significant bit positions. If n2 is negative, n1 is shifted right. Sign extension is to be consistent with the processor's arithmetic shift.

B/BUF -- 1024

A constant leaving 1024, the number of bytes per block buffer. "bytes-per-buffer"

BELL

Activate a terminal bell or noise-maker as appropriate to the device in use.

BL -- n 176

Leave the ASCII character value for blank (decimal 32). "b-1"

BLANKS addr n -- 152

Fill an area of memory over n bytes with the value for ASCII blank, starting at addr. If n is less than or equal to zero, take no action.

C, n --

Store the low-order 8 bits of n at the next byte in the dictionary, advancing the dictionary pointer. "c-comma"

CHAIN

Used in the form:

CHAIN <name>

Connect the CURRENT vocabulary to all definitions that might be entered into the vocabulary <name> in the future. The CURRENT vocabulary may not be FORTH or ASSEMBLER. Any given vocabulary may only be chained once, but may be the object of any number of chainings. For example, every user-defined vocabulary may include the sequence:

CHAIN FORTH

COM n1 -- n2

Leave the one's complement of n1.

CONTINUED n --

Continue interpretation at block n.

CUR —— addr

A variable pointing to the physical record number before which the tape is currently positioned. REWIND sets CUR=1.

DBLOCK d —— addr

Identical to BLOCK but with a 32-bit block unsigned number. "D-block"

DPL —— addr

A variable containing the number of places after the fractional point for output conversion. If DPL contains zero, the last character output will be a decimal point. No point is output if DPL contains a negative value. DPL may be set explicitly, or by certain output words, but is unaffected by number input. "d-p-l"

DUMP addr n —— 123

List the contents of n addresses starting at addr. Each line of values may be preceeded by the address of the first value.

EDITOR I,172

The name of the editor vocabulary. When this name is executed, EDITOR is established as the CONTEXT vocabulary.

END I,C,224

A synonym for UNTIL.

ERASE addr n —— 182

Fill an area of memory over n bytes with zeros, starting at addr. If n is zero or less, take no action.

FLD —— addr

A variable pointing to the field length reserved for a number during output conversion. "f-l-d"

FLUSH

A synonym for SAVE-BUFFERS

H. n ——

Output n as a hexadecimal integer with one trailing blank. The current base is unchanged. "h-dot"

HEX 162

Set the numeric input-output conversion base to sixteen.

I' —— n C

Used within a colon-definition executed only from within a DO-LOOP to return the corresponding loop index. "i-prime"

IFEND

Terminate a conditional interpretation sequence begun by IFTRUE.

IFTRUE flag ——

Begin an

IFTRUE ... OTHERWISE ... IFEND

conditional sequence. These conditional words operate like

IF ... ELSE ... THEN

except that they cannot be nested, and are to be used only during interpretation. In conjunction with the words [and] they may be used within a colon-definition to control compilation, although they are not to be compiled.

INDEX nl n2 ——

Print the first line of each screen over the range {nl..n2}. This displays the first line of each screen of source text, which conventionally contains a title.

INTERPRET

Begin interpretation at the character indexed by the contents of >IN relative to the block number contained in BLK, continuing until the input stream is exhausted. If BLK contains zero, interpret characters from the terminal input buffer.

K —— n C

Within a nested DO-LOOP, return the index of the second outer loop.

LAST —— addr

A variable containing the address of the beginning of the last dictionary entry made, which may not yet be a complete or valid entry.

LINE n —— addr

Leave the address of the beginning of line n for the screen whose number is contained in SCR. The range of n is {0..15}.

LINELOAD nl n2 ——

Begin interpretation at line nl of screen n2.

LOADS n —

A defining word used in the form:

n LOADS <name>

When <name> is subsequently executed, block n will be loaded.

MAP0 — addr

A variable pointing to the first location in the tape map.

MASK n1 — n2

Leave a mask of n1 most significant bits if n1 is positive, or n least significant bits if n1 is negative.

MS n —

Delay for approximately n milliseconds.

NAND n1 n2 — n3

The one's complement of the logical and of n1 and n2.

NOR n1 n2 — n3

The one's complement of the logical or of n1 and n2.

NUMBER addr — n

Convert the count and character string at addr, to a signed 32-bit integer, using the current base. If numeric conversion is not possible, an error condition exists. The string may contain a preceding negative sign.

O. n —

Print n in octal format with one trailing blank. The value in BASE is unaffected. "o-dot"

OCTAL

Set the number conversion base to 8.

OFFSET — addr 128

A variable that contains the offset added to the block number on the stack by BLOCK to determine the actual physical block number.
The user must add any desired offset when utilizing BUFFER.

OTHERWISE

An interpreter-level conditional word. See IFTRUE.

PAGE

Clear the terminal screen or perform an action suitable to the output device currently active.

READ-MAP

Read to the next file mark on tape constructing a correspondence table in memory (the map) relating physical block position to logical block number. The tape should normally be rewound to its load point before executing READ-MAP.

REMEMBER

A defining word used in the form:

REMEMBER <name>

Defines a word which, when executed, will cause <name> and all subsequently defined words to be deleted from the dictionary. <name> may be compiled into and executed from a colon definition. The sequence

DISCARD REMEMBER DISCARD

provides a standardized preface to any group of transient word definitions.

REWIND

Rewind the tape to its load point, setting CUR=1.

ROTATE n1 n2 -- n3

Rotate the value n1 left n2 bits if n2 is positive, right n2 bits if n2 is negative. Bits shifted out of one end of the cell are shifted back in at the opposite end.

S0 -- addr

Returns the address of the bottom of the stack, when empty. "s-zero"

SET n addr --

A defining word used in the form:

n addr SET <name>

Defines a word <name> which, when executed, will cause the value n to be stored at address.

SHIFT n1 n2 -- n3

Logical shift n1 left n2 bits if n2 in positive, right if n2 is negative. Zeros are shifted into vacated bit positions.

SP@ — addr 214

Return the address of the top of the stack, just before SP@ was executed. "s-p-fetch"

TEXT c —

Accept characters from the input stream, as for WORD, into PAD, blank-filling the remainder of PAD to 64 characters.

THRU n1 n2 —

Load consecutively the blocks from n1 through n2.

U.R unl n2 — 216

Output unl as an unsigned number right justified in a field n2 characters wide. If n2 is smaller than the characters required for n1, no leading spaces are given. "u-dot-r"

USER n —

A defining word used in the form:

n USER <name>

which creates a user variable <name>. n is the cell offset within the user area where the value for <name> is stored. Execution of <name> leaves its absolute user area storage address.

VLIST

List the word names of the CONTEXT vocabulary starting with the most recent definition.

WHERE

Output information about the status of FORTH, (e.g., after an error abort). Indicate at least the last word compiled and the last block accessed.

\ LOOP n — I,C

A DO-LOOP terminating word. The loop index is decremented by n and the loop terminated when the resultant index becomes equal to or less than the limit. Magnitude logic is used, and n must be positive. "down-loop"

FORTH-79 HANDY REFERENCE

Stack inputs and outputs are shown; top of stack on right. See operand key at bottom.

STACK MANIPULATION

DUP	(n → n n)	Duplicata top of stack.
DROP	(n →)	Discard top of stack.
SWAP	(n1 n2 → n2 n1)	Exchange top two stack items.
OVER	(n1 n2 → n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 → n2 n3 n1)	Rotate third item to top. "rote"
PICK	(n1 → n2)	Copy n1-th item to top. (Thus 1 PICK = DUP , 2 PICK = OVER)
ROLL	(n →)	Rotate n-th item to top. (Thus 2 ROLL = SWAP , 3 ROLL = ROT)
?DUP	(n → n (n))	Duplicate only if non-zero. "query-dup"
>R	(n →)	Move top item to "return stack" for temporary storage (use caution). "to-r"
R>	(→ n)	Retrieve item from return stack. "r-from"
R@	(→ n)	Copy top of return stack onto stack. "r-fetch"
DEPTH	(→ n)	Count number of items on stack.

COMPARISON

<	(n1 n2 → flag)	True if n1 less than n2. "less-than"
=	(n1 n2 → flag)	True if top two numbers ara equal. "equals"
>	(n1 n2 → fleg)	True if n1 greeter than n2. "graater-than"
0<	(n → flag)	True if top number negative. "zaro-less"
0=	(n → flag)	True if top number zero. (Equivalent to NOT) "zero-equals"
0>	(n → flag)	True if top number graater than zaro. "zero-greater"
D<	(d1 d2 → flag)	True if d1 less than d2. "d-lass-than"
U<	(un1 un2 → flag)	Compare top two items as unsigned integers. "u-less-than"
NOT	(fleg → ¬flag)	Revarse truth value. (Equivalant to 0=)

ARITHMETIC AND LOGICAL

+	(n1 n2 → sum)	Add. "plus"
D+	(d1 d2 → sum)	Add double-precision numbers. "d-plus"
−	(n1 n2 → diff)	Subtract (n1−n2). "minus"
1+	(n → n+1)	Add 1 to top number. "one-plus"
1−	(n → n−)	Subtract 1 from top number. "one-minus"
2+	(n → n+2)	Add 2 to top number. "two-plus"
2−	(n → n−2)	Subtract 2 from top number. "two-minus"
*	(n1 n2 → prod)	Multiply. "times"
/	(n1 n2 → quot)	Divide (n1/n2). (Quotient rounded toward zero) "divida"
MOD	(n1 n2 → rem)	Modulo (*i.e.*, remainder from division n1/n2). Remeinder has same sign as n1. "mod"
/MOD	(n1 n2 → rem quot)	Divide, giving remainder and quotiant. "divide-mod"
*/MOD	(n1 n2 n3 → rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermedlate. "times-divide-mod"
*/	(n1 n2 n3 → quot)	Like */MOD , but give quotient only, rounded toward zero. "times-divide"
U*	(un1 un2 → ud)	Multiply uneigned numbers, leaving unsigned double-precision result. "u-times"
U/MOD	(ud un → urem uquot)	Divide double number by single, giving remainder and quotient, all unsigned. "u-divide-mod"
MAX	(n1 n2 → max)	Leave greeter of two numbers. "max"
MIN	(n1 n2 → min)	Leave lesser of two numbere. "min"
ABS	(n → |n|)	Absoluta value. "absolute"
NEGATE	(n → −n)	Leave two's complement.
DNEGATE	(d → −d)	Leava two's complement of double-precision number. "d-negate"
AND	(n1 n2 → and)	Bitwise logical AND.
OR	(n1 n2 → or)	Bitwise logical OR.
XOR	(n1 n2 → xor)	Bitwise logical exclusive-OR. "x-or"

MEMORY

@	(addr → n)	Replace address by number at address. "fetch"
!	(n addr →)	Store n at addr. "store"
C@	(addr → byte)	Fetch least significant byte only. "c-fetch"
C!	(n addr →)	Store least signficant byte only. "c-store"
?	(addr →)	Display number at address. "question-mark"
+!	(n addr →)	Add n to number at addr. "plus-store"
MOVE	(addr1 addr2 n →)	Move n numbers starting at addr1 to memory starting at addr2, if n>0.
CMOVE	(addr1 addr2 n →)	Move n bytes starting at addr1 to memory starting at addr2, if n>0. "c-move"
FILL	(addr n byte →)	Fill n bytes in memory with byte beginning at addr, if n>0.

CONTROL STRUCTURES

DO . . . LOOP	do: (end+1 start →)	Set up loop, given index range.
I	(→ indax)	Place current loop index on data stack.
J	(→ indax)	Return index of next outer loop in same definition.
LEAVE	(→)	Terminate loop at next LOOP or +LOOP , by setting limit equal to index.
DO . . . +LOOP	do: (limit start →) +loop: (n →)	Like DO . . . LOOP , but adds stack value (insteed of always 1) to index. Loop terminates when index is graater than or equal to limit (n>0), or when index is less than limit (n<0). "plus-loop"
IF . . . (true) . . . THEN	if: (flag →)	If top of stack true, execute.
IF . . . (trua) . . . ELSE . . . (false) . . . THEN	if: (flag →)	Seme, but if false, execute ELSE clause.
BEGIN . . . UNTIL	until: (flag →)	Loop back to BEGIN until true et UNTIL .
BEGIN . . . WHILE . . . REPEAT	while: (flag →)	Loop while trua at WHILE; REPEAT loops unconditionally to BEGIN . When false, continue after REPEAT .
EXIT	(→)	Terminate execution of colon definition. (Mey not be used within DO . . . LOOP)
EXECUTE	(addr →)	Execute dictionary entry et compilation address on stack (*e.g.*, address returned by FIND).

Operand key:

n, n1, . . .	16-bit signed numbers	d, d1, . . .	32-bit signed numbers	addr, addr1, . . .	addreases	char	7-bit ascii character value
		u	unsigned	byte	8-bit byta	flag	boolean llag

TERMINAL INPUT-OUTPUT

CR	(→)	Do a carriage return and line feed. "c-r"
EMIT	(char →)	Type ascii valua from stack.
SPACE	(→)	Type one space.
SPACES	(n →)	Type n spaces, if n>0.
TYPE	(addr n →)	Type string of n characters beginning at addr, if n>0.
COUNT	(addr → addr+1 n)	Change address of string (prafixed by length byta at addr) to TYPE form.
-TRAILING	(addr n1 → addr n2)	Reduce charecter count of string at addr to omit trailing blanks. "dash-trailing"
KEY	(→ char)	Read key and leave ascii valua on stack.
EXPECT	(addr n →)	Raad n characters (or until carriage return) from terminal to address, with null(s) at end.
QUERY	(→)	Read line of up to 80 characters from terminal to input buffer.
WORD	(char → eddr)	Read next word from input stream using char as delimiter, or until null. Leave addr of length byte.

NUMERIC CONVERSION

BASE	(→ addr)	System variabla containing radix for numeric conversion.
DECIMAL	(→)	Set decimal number base.
.	(n →)	Print number with one trailing blank and sign if negative. "dot"
U.	(un →)	Print top of stack as unsigned number with one trailing blank. "u-dot"
CONVERT	(d1 eddr1 → d2 addr2)	Convert string at eddr1+1 to double number. Add to d1 laaving sum d2 and addr2 of first non-digit.
<#	(→)	Start numaric output string conversion. "less-sharp"
#	(ud1 → ud2)	Convert next digit of unsigned doubla number and add character to output string. "sharp"
#S	(ud → 0 0)	Convart all significant digits of unsigned doubla number to output string. "sharp-s"
HOLD	(char →)	Add ascii cher to output string.
SIGN	(n →)	Add minus sign to output string if n<0.
#>	(d → addr n)	Drop d and tarminate numeric output string, leaving addr and count for TYPE . "sharp-greater"

MASS STORAGE INPUT/OUTPUT

LIST	(n →)	List scraan n and set SCR to contain n.
LOAD	(n →)	Interpret screen n, then resuma intarpratation of the currant input straam.
SCR	(→ eddr)	Systam variable containing scraen number most recantly listed.
BLOCK	(n → addr)	Leave memory address of block, reading from mass storage if necessary.
UPDATE	(→)	Mark last block raferanced as modified.
BUFFER	(n → eddr)	Leave addr of e free buffer, assigned to block n; writa previous contents to mass storage if UPDATEd.
SAVE-BUFFERS	(→)	Write all UPDATEd blocks to mass storaga.
EMPTY-BUFFERS	(→)	Mark all block buffers as empty, without writing UPDATEd blocks to mass storage.

DEFINING WORDS

: xxx	(→)	Begin colon definition of xxx . "colon"
;	(→)	End colon dafinition. "semi-colon"
VARIABLE xxx	(→) xxx: (→ addr)	Create a two-byte variable named xxx ; returns address when executed.
CONSTANT xxx	(n →) xxx: (→ n)	Create a constant named xxx with velue n; returns value when axecuted.
VOCABULARY xxx	(→)	Creata a vocabulary named xxx ; becomes CONTEXT vocabulary when axecuted.
CREATE . . . DOES>	does: (→ addr)	Used to craete e new defining word, with axecution-tima routina in high-level FORTH. "does"

VOCABULARIES

CONTEXT	(→ addr)	System variable pointing to vocabulary where word names ara searched for.
CURRENT	(→ sddr)	System variable pointing to vocabulary whara new dafinitions are put.
FORTH	(→)	Main vocabulary, contained in all other vocabularias. Execution of FORTH sets context vocabulary.
DEFINITIONS	(→)	Sets CURRENT vocabulary to CONTEXT.
' xxx	(→ addr)	Find eddress of xxx in dictionary; if used in definition, compile address. "tick"
FIND	(→ addr)	Leave compilation address of next word in input stream. If not found in CONTEXT or FORTH, leave 0.
FORGET xxx	(→)	Forget all definitions back to and including xxx , which must be in CURRENT or FORTH.

COMPILER

,	(n →)	Compile e number into the dictionary. "comma"
ALLOT	(n →)	Add two bytes to the parametar field of tha most recently-defined word.
."	(→)	Print messaga (tarminated by "). If used in definition, print when exacuted. "dot-quote"
IMMEDIATE	(→)	Mark last-dafined word to be exacuted when encountarad in e definition, rathar than compiled.
LITERAL	(n →)	If compiling, save n in dictionary, to be raturned to stack when dafinition is axecuted.
STATE	(→ addr)	System variabla whose value is non-zero when compilation is occurring.
[	(→)	Stop compiling input taxt and begin axecuting. "laft-brecket"
]	(→)	Stop axacuting input text and begin compiling. "right-bracket"
COMPILE	(→)	Compile the eddress of the next non-IMMEDIATE word into the dictionary.
[COMPILE]	(→)	Compile the following word, even if IMMEDIATE. "bracket-compile"

MISCELLANEOUS

(	(→)	Begin comment, tarminated by) on same line or screen; space efter (. "paren", "close-paren"
HERE	(→ addr)	Leave addrass of naxt available dictionary location.
PAD	(→ addr)	Leave addrass of a scratch area of at laast 64 bytes.
>IN	(→ addr)	Systam variabla containing character offsat into input buffar; used, *e.g.*, by WORD. "to-in"
BLK	(→ addr)	System variabla containing block number currently being intarpratad, or 0 if from tarminal. "b-l-k"
ABORT	(→)	Clear data and raturn stacks, set exacution mode, raturn control to tarminal.
QUIT	(→)	Like ABORT , axcept does not clear data stack or print any massage.
79-STANDARD	(→)	Verify that system conforms to FORTH-79 Standard.